Management Sciences Research Report No. MSRR 566

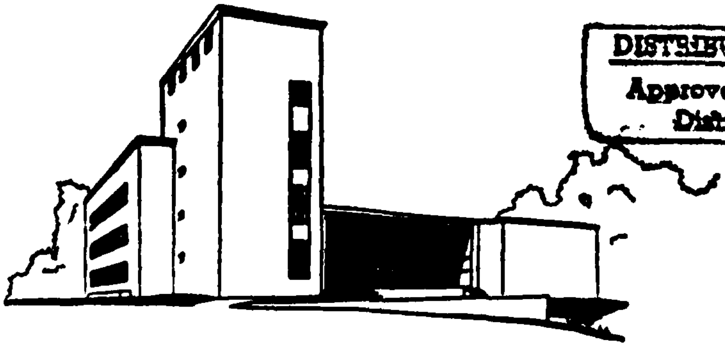
An Exact Two-Matching Based Branch and Bound Algorithm for the Symmetric Traveling Salesman Problem

February, 1991

Carnegie Mellon University

PITTSBURGH, PENNSYLVANIA 15213

GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION
WILLIAM LARIMER MELLON, FOUNDER



(1)

W.P. #1991-21

Management Sciences Research Report No. MSRR 566

AN EXACT TWO-MATCHING BASED BRANCH AND BOUND ALGORITHM FOR THE SYMMETRIC TRAVELING SALESMAN PROBLEM

FEBRUARY, 1991

W.P. No. 1991-21

Management Sciences Research Report No. MSRR 566

AN EXACT TWO-MATCHING BASED BRANCH AND BOUND ALGORITHM FOR THE SYMMETRIC TRAVELING SALESMAN PROBLEM

D. L. Miller

Central Research & Development Department
E. I. du Pont de Nemours and Company
Wilmington, DE 19880

J. F. Pekny†

School of Chemical Engineering
Purdue University
West Lafayette, IN 47907

G. L. Thompson‡

Graduate School of Industrial Administration
Carnegie Mellon University
Pittsburgh, PA 15213

† This work has been supported by the Engineering Design Research Center at Carnegie Mellon University, an NSF Engineering Research Center.

‡ Work supported in part by the Office of Naval Research, under Contract No. N00014-85-K-0198 NR 047-048, and in part by the Department of Energy, under Contract No. DE-FG02-85ER13396-1H121A. Reproduction in whole or in part is permitted for any purpose of the U. S. Government.

Management Sciences Research Group
Graduate School of Industrial Administration
Carnegie Mellon University
Pittsburgh, PA 15213

An Exact Two-Matching Based Branch and Bound Algorithm for the Symmetric Traveling Salesman Problem

D. L. Miller

Central Research & Development Department
E. I. du Pont de Nemours and Company
Wilmington, DE 19880

J. F. Pekny

School of Chemical Engineering
Purdue University
West Lafayette IN 47907

G. L. Thompson

Graduate School of Industrial Administration
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

We describe an algorithm for the symmetric traveling salesman problem (TSP) based on a bipartite two-matching lower bounding technique. The lower bound is strengthened by using the bipartite two-matching as the basis for a heuristic algorithm for the dual of integer two-matching. We use this dual as a lower bound for the symmetric traveling salesman problem in a branch and bound algorithm. Results are presented for random symmetric TSPs with up to 3000 cities. On Euclidean problems the two-matching bound is weaker than on random problems and algorithm performance deteriorates as a result. The algorithm successfully solves 11 of 15 Euclidean problems from the literature with sizes ranging from 17 to 99 cities.

1. Introduction

Although the traveling salesman problem (TSP) is NP-complete, some success has been achieved recently in obtaining exact solutions for certain problem classes. Random asymmetric problems with up to 500,000 cities have been solved using assignment problem (AP) lower bounding techniques [1-3]. A euclidean problem with more than 2300 cities was solved to optimality using lower bounds from an LP relaxation strengthened by cutting planes [4].

The two-matching problem is the natural analog of the AP for the symmetric TSP and it therefore makes sense to investigate its performance as a lower bounding technique. Bellmore and Malone [5] experienced some success with the two-matching relaxation for the symmetric TSP but were limited to investigating small problems by the available computer technology and lack of an effective two-matching algorithm. Two-matching is a polynomial class problem and may be solved using general LP codes [6] or codes based on the work of Edmonds [7]. In section 2, we present a method for obtaining bounds close to that of two-matching using a much faster special purpose algorithm. We relax the blossom constraints [8] required for integer two-matching and transform the relaxation to a bipartite two-matching problem which may be solved as a capacitated network flow problem. This bound is then strengthened using a primal-dual algorithm, producing a primal two-matching, which may be sub-optimal, and a dual solution whose cost is a valid lower bound on the cost of an optimal two-matching. The gap between the primal and dual objective functions may be removed using a general LP solver as a post processing step, but in practice the computational effort may not be justified. In section 3 we describe a branch and bound algorithm for the symmetric TSP, based on the lower bounds provided by our approximate dual two-matching algorithm. Section 4 contains computational results on random symmetric TSPs and on euclidean problems taken from the literature. Section 5 presents the conclusions.

2. Primal and Dual Two-Matching Problem Formulation

Two-matching, which is a special case of b-matching with upper bounds, may be formulated as an integer program on an undirected graph $G=(V,E)$ with edge weights c_{ij} and binary edge variables x_{ij} as follows:

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{1}$$

subject to:

$$\sum_{j:(i,j)\in E} x_{ij} = 2, \quad \forall i\in V \tag{2}$$

$$x_{ij}\in \{0,1\} \quad \forall (i,j)\in E \tag{3}$$

If integrality is replaced by the following constraint, the resulting linear program admits only half integral solutions, i.e. $x_{ij}\in \{0,\tfrac{1}{2},1\}$.

$$0\le x_{ij}\le 1 \quad \forall i,j\in E \tag{4}$$

The linear program of Equations (1,2,4) may be rapidly solved via a capacitated network flow algorithm on a bipartite auxiliary graph as described in [9]. Obtaining an optimal solution to Equations (1-3) requires eliminating the half integral x_{ij} at minimal cost. Following [10], the half integral x_{ij} may be interpreted with respect to graph G as forming cycles of odd and even numbers of edges. Equation (5) defines the *blossom* constraints which eliminate the half integral solutions [6].

$$\forall R \in \Re,\ T\subseteq\delta(R), |T|\ odd: \sum_{(i,j)\in\sigma(R)} x_{ij} + \sum_{(i,j)\in T} x_{ij} \le \tfrac{1}{2}[2|R|+|T|-1] \tag{5}$$

where for $w\subseteq V$:

$\delta(w) = \{(i,j)\in E \mid i\in w, j\notin w\}$

$\sigma(w) = \{(i,j)\in E \mid i\in w, j\in w\}$

$\Re = \{R\subseteq V \mid |R|\ge 3\}$

An integer two-matching may be found in a straightforward manner by iteratively adding violated blossom constraints to the LP until an integer solution is obtained [6]. The problem size which may be solved using this approach is limited by the use of general purpose LP codes which don't exploit problem specific structure. Here we introduce a fast, albeit approximate method for obtaining the two-matching lower bounds in the symmetric TSP. Dualizing the constraints in (5), yields the following dual formulation for two-matching:

$$Maximize\, \Theta(y_i, y_{R,T}) \tag{6}$$

$$\Theta(y_i, y_{R,T}) = \min_{0 \le x_{ij} \le 1} \left[2 \sum_{i \in V} y_i - \sum_{\forall R,T} y_{R,T} \Phi_{R,T} + \sum_{(i,j) \in E)} \bar{c}_{ij} x_{ij} \right]$$

subject to:

$y_{R,T} \geq 0$

where

$$\bar{c}_{ij} = c_{ij} - y_i - y_j + \sum_{(R,T) \in \Gamma_{ij}} y_{R,T} \tag{7}$$

and

$$\Gamma_{ij} = \{ (R,T) \,|\, R \in \Re,\ T \subseteq \delta(R),\ |T|\ odd,\ (i,j) \in \sigma(R) \cup T \}$$

$$\Phi_{R,T} = \tfrac{1}{2}[2|R| + |T| - 1]$$

$$\sum_{\forall R,T} = \sum_{\forall R \in \Re, T \subseteq \delta(R), |T|\ odd}$$

The complementary slackness conditions implied by the dual are as follows:

$$\bar{c}_{ij} > 0 \rightarrow x_{ij} = 0 \tag{8}$$

$$\bar{c}_{ij} < 0 \rightarrow x_{ij} = 1 \tag{9}$$

$$\bar{c}_{ij} = 0 \rightarrow x_{ij} = 0 \text{ or } x_{ij} = 1 \tag{10}$$

$$y_{R,T} > 0 \rightarrow \sum_{(i,j) \in \sigma(R)} x_{ij} + \sum_{(i,j) \in T} x_{ij} = \tfrac{1}{2}[2|R| + |T| - 1] \tag{11}$$

The dual objective value $\Theta(y_i, y_{R,T})$ is a lower bound on the cost of an optimal two-matching and thus on the optimal cost of a TSP solution.

3. A Fast Heuristic Algorithm for the Two-Matching Dual

Our algorithm for solving (6) consists of three phases, (1) solving Equations (1,2,4), (2) even cycle elimination, (3) odd cycle elimination.

1. LP Solution Using a Bipartite Auxiliary Graph

Equations (1,2,4) may be solved by any general purpose LP solver. However, on a suitably defined directed bipartite graph, the problem may be solved using much faster capacitated network flow algorithms. Given $G=(V,E)$, consider the related graph $B=(S,T,A)$ with vertex sets $S=V$, $T=V$ and arc set $A=\{(i,j): i\in S, j\in T, (i,j)\in E\}$. Each vertex in S is a source of two units of flow and each vertex in T is a sink for two units, each arc having a maximum capacity of one unit of flow. If the solution is interpreted as a bipartite two-matching, each vertex in S and T will be incident to two arcs. The bipartite two-matching interpretation may be used to construct an optimal solution to Equations (1,2,4) using the following three rules: (1) If arcs (i,j) and (j,i) are present, then x_{ij}, (i,j)∈ E, is one in the LP solution; (2) If arc (i,j) or (j,i) is present, then x_{ij}, (i,j)∈ E is one half in the LP solution; (3) All other x_{ij}, (i,j)∈ E are zero. If the bipartite two-matching solution on B is symmetric, i.e. arc (i,j) is present if and only if arc (j,i) is present, an optimal integer two-matching is found on G. If not, the half integral variables of the LP solution form a set of cycles on the edges of G. The removal of these cycles is required to find an optimal integer two-matching.

2. Even cycle elimination

Cycles of even length and pairs of odd length cycles which are not disjoint may be trivially removed without altering the cost of the solution, see [9], Figure 1, and Figure 2a,2b.

3. Disjoint odd cycle elimination

The removal of disjoint odd cycles is accomplished by finding zero reduced cost walks, whose edges alternate between those in and out of the matching, connecting a pair of odd cycles. Once such a walk is found, the matching may be altered by complementing the variables corresponding to the edges in the walk. Both odd cycles are eliminated by alternately setting variables in each odd cycle to zero and one starting at the vertex at which the walk is incident in such a way as to obtain a two-matching (see Figure 2c,2d). The alternating walks are found by growing an alternating tree rooted at an odd cycle. The labeling technique used here is taken from Anstee [10] who describes a method for building such alternating trees for the unweighted b-matching problem. All vertices in the tree possess alternating walks back to the root odd cycle. Vertices whose parent edge is in or out of the matching, receive T and S labels respectively. Because graph G is not bipartite, blossoms may form and these are collapsed into pseudo-vertices as described in [10]. If the tree becomes Hungarian, the dual variables are increased to bring additional arcs into the tree. If the alternating walks required for odd cycle elimination are found without increases to the dual variables, we find an optimal two-matching at the same cost as the LP relaxation. If dual variable increases are required, the integer two-matching cost is elevated above that of the LP relaxation.

The method outlined below requires that the dual variables associated with certain blossom constraints be made nonzero. Complementary slackness requires that blossom constraints with nonzero y_{RT} be satisfied at equality rather than inequality (see Equation (11)). Proper construction of the alternating walk guarantees that complementary slackness is satisfied for the blossom constraints associated with pseudovertices formed in the alternating tree. However, consider the case when an alternating walk intersects a pseudovertex from a tree used to eliminate a previously existing pair of odd cycles. If the y_{RT} for that pseudovertex is nonzero, the transfer may alter the matching so that the blossom constraint is satisfied with inequality. This leads to a duality gap between the cost of the primal and dual solutions, although the dual remains a valid lower bound. Although this gap is undesirable, in practice it is often small or non-existent so that the lower bounds are still strong enough to solve many nontrivial symmetric TSPs. Naturally, this gap may be removed by using a general purpose LP solver with the primal solution as a starting point. Derigs avoids a duality gap in a 1-matching algorithm by locally adjusting the dual variables whenever an alternating walk intersects an existing blossom with associated nonzero dual [11]. The blossom constraints are more complicated for two-matching and apparently no analogous local method for dual variable adjustment exists.

4. Algorithm for Removing Odd Directed Cycles

We now precisely describe an algorithm for removing disjoint pairs of odd cycles until an integer two-matching on G is constructed. In the description, set L refers to those vertices from which further alternating tree growth may occur.

while(an odd cycle remains)
begin

Find the smallest odd cycle and form its members into a pseudovertex fixing the sets R and T. If no such cycle exists, an integer two-matching on G has been found. Give all members of the cycle S and T labels. Add the members to the set L.

while(set L is not empty)
begin

Remove a vertex u from L.

Perform the following if u is S labeled: Scan the S labeled vertex u. For each w with (u,w) in the matching and $\bar{c}_{uw}=0$, form a new pseudovertex using edge (u,w) if w is S labeled (see Pseudovertex Formation below). If w is not S or T labeled, edge (u,w) is in the matching, and $\bar{c}_{uw} = 0$, add (u,w) to the tree, place w in L, and give w a T label. If w is a member of an odd cycle, use the alternating walk to eliminate the odd cycles at each end and repeat from the beginning of the outermost while loop.

Perform the following if u is T labeled: Scan the T labeled vertex u. For all vertices $w \neq u$ such that (u,w) is not in the matching and $\bar{c}_{uw}=0$, form a new pseudovertex if w is T labeled. If w is not S or T labeled and $\bar{c}_{uw} = 0$, add edge (u,w) to the tree, place w in L, and give w an S label. If w is a member of an odd cycle, use the alternating walk to eliminate the odd cycles at each end and repeat from the beginning of the outermost while loop.

end

An f-barrier has been found. Find an edge (u,w) with minimal absolute value reduced cost such that the edge can either expand the tree or form a new pseudovertex. Consider edges in which u is in the tree and w is not (these lead to expansion of the tree) and edges in which both u and w are in the tree (these form new pseudovertices). Let τ be the minimum absolute value reduced cost. Adjust the dual cost by τ. For all vertices i adjust y_i by $-\tau$ if i is S labeled and not T labeled and by τ if i is T labeled but not S labeled. For all outermost pseudo-vertices, adjust y_{RT} by $2*\tau$ and y_i by τ for $i \in R$. Check for further pseudovertex formation caused by the dual variable update. For all new zero cost intra-tree edges (u,w) form a new pseudovertex. For all edges (u,w) not in the matching, with u T labeled, w not in the tree, and $\bar{c}_{uw} = 0$, give w an S label. For all edges (u,w) in the matching, with u S labeled, w not in the tree, and $\bar{c}_{uw} = 0$, give w an T label.
end

We now explain pseudovertex formation and odd cycle elimination in more detail:

Pseudovertex Formation

Pseudovertices may be formed by adding edge (u,w) to the alternating tree under either of the following conditions: (i) u and w are S labeled and the edge (u,w) is in the matching, (ii) u and w are T labeled and the edge (u,w) is not in the matching. Furthermore, u and w must not already be in the same pseudovertex and reduced cost of the edge $\bar{c}_{uw}$ must be zero. When edge (u,w) forms a new pseudovertex, all vertices in the cycle created by (u,w) are incorporated into the new pseudovertex. Note that some of these vertices may themselves be pseudovertices. All actual (non-pseudo) vertices contained in the pseudovertex form the R set. For each $u \in R$, there is a unique vertex $w \notin R$, (u,w) in the matching. These edges (u,w) form the T set. All vertices in R receive both S and T labels.

Odd cycle elimination

Let (i,j) be the edge of an alternating walk incident to an odd cycle, with vertex j a member of the cycle. Let the $P = \{P_0, P_1, \cdots, P_{2l+1}\}$ denote the vertices in the cycle, with $P_0 = P_{2l+1}$ = vertex j. If edge (i,j) is originally in the matching, then the cycle gains parity, i.e. following augmentation edges (P_0, P_1) and (P_{2l}, P_{2l+1}) are in the matching, and the remaining edges in the odd cycle alternately are out and in the matching. Otherwise, the cycle loses parity and following augmentation edges (P_0, P_1) and (P_{2l}, P_{2l+1}) are out of the matching, and the remaining edges in the odd cycle alternately are in and out of the matching.

5. Branch and Bound Algorithm

An integer programming formulation for the symmetric TSP is given by (1-3) plus the following subtour elimination constraints:

$$\sum_{i \in S} \sum_{j \in S, j > i} x_{ij} \le |S| - 1 \quad \forall S \subset V, 3 \le |S| \le n-1 \tag{12}$$

We developed a branch and bound algorithm by dropping the subtour elimination constraints and utilizing the dual objective function from the previous section as a lower bound. The constraints in Equation (12) are enforced using branching rules as described in [12, 13]. Upper bounds are provided by patching the primal two-matching into a single tour using the algorithm of Karp [14].

The speed of the computation is enhanced using a sparse matrix method similar to that described in [2]. We form a sparse cost matrix by discarding all elements larger than some threshold λ. The optimality of the solution relative to the fully dense matrix is checked after the solution is obtained. Let v(STSP) and v(root) be the solution to the TSP obtained from the sparse cost matrix and the value of the root node bipartite two-matching lower bound respectively. Since a positive reduced cost of an edge (i,j) is a lower bound on the cost increase caused by including edge (i,j), it suffices to check that:

$$\lambda + 1 - y_i - y_j > v(STSP) - v(root) \quad \forall i,j \tag{13}$$

Equation (13) requires an $O(N^2)$ check but may be replaced by the less stringent condition:

$$\lambda + 1 - y_i - y_{max} > v(STSP) - v(root) \quad \forall i,j \tag{14}$$

where y_{max} is the largest y_i. If this condition fails, the problem is resolved with a larger value of λ.

6. Computational Results

We tested the algorithm on symmetric TSPs with random cost matrices and on euclidean problems. Table (1) shows the results for random symmetric problems with cost matrix elements in the range [0-n]. We solved five problems of each size for n=100, 500, 1000, 2000, and 3000 using a Sun Microsystem 4/330 deskside computer. Random symmetric problems rarely possess more than six disjoint odd cycles and phase three of the algorithm is usually able to remove them with no duality gap. Of the 25 problems in Table (1), only one had a duality gap at the root node of the branch and bound tree. The two-matching lower bound was in excess of 99% of the optimal tour cost on all random problems tested. These results are not surprising as random problems are known to be easier than euclidean problems. The speed of this algorithm, however, on random problems seems to be superior to those based on general LP solvers. Grötschel and Holland report the solution of a 1000 city random problem in 0.5 hours on an IBM 3081D [15]. As Table (1) shows, our algorithm is faster on similarly structured random problems†

This algorithm is less suited for Euclidean problems for two reasons. First, the number of disjoint odd cycles is larger than for similarly sized random symmetric problems, and a duality gap frequently persists. The second reason for diminished performance on Euclidean problems is the weakness of the two-matching bound. To compare the relative importance of these two factors, consider the 532 city problem of Padberg and Rinaldi [4]. The bipartite two-matching bound is 26620.5. Elimination of half integral variable values via our heuristic increases the bound to the dual cost of 26823 while the primal cost is 26916. The duality gap here is 0.35% while the gap between the dual and the optimal tour cost is much larger, 3.2%. Thus the difficulty in solving euclidean problems is attributable more to the weakness of the two-matching bounds than to the duality gap.

We tested the algorithm on standard euclidean problems available from the literature. We used a suite of test problems, TSPLIB, which is available on an academic computer network [16]. Table (2) shows algorithm performance on all problems in TSPLIB with about one hundred cities or less. The algorithm was able to solve eleven of the fifteen test problems. The largest problem the algorithm solved was 101 cities. For some problems the solution time is reasonable, but for several others it is unacceptably long.

7. Conclusions

We have presented a dual formulation for the two-matching problem along with a heuristic solution algorithm based on primal-dual methods. A duality gap arises because the heuristic does not necessarily satisfy complementary slackness conditions, although this deficiency may be removed using a general purpose LP solver as a post processing step. For random symmetric cost matrices, the gap is small (frequently zero) and the heuristic closely approximates the cost of an optimal two-matching, thus providing reasonable lower bounds for some symmetric TSPs. Random symmetric problems possessing up to three thousand cities may be solved in a little over four minutes using a Sun 4/330.

On euclidean problems, two-matching alone does not provide a bound of sufficient strength to solve large problems. For this reason, the algorithm presented here is not competitive with cutting plane based methods for euclidean TSPs possessing more than a few dozen cities. Nevertheless, our primal-dual method for obtaining near optimal two-matching bounds is much faster than general purpose LP codes. If further work can eliminate the duality gap and strengthen the two-matching bound, the speed of special purpose lower bounding algorithms may yield an exact symmetric TSP algorithm which is at least competitive with cutting plane based methods that utilize general purpose LP solvers.

---

† An IBM 3081D is approximately 50% faster than a Sun Microsystem 4/330.

Random Symmetric Problems				
problem size	cases	bound strength	number of search tree vertices	solution time (sec)
100	5	0.998652	7.6	0.696
500	5	0.999488	97.8	32.31
1000	5	0.999482	178.2	162.116
2000	5	0.999677	158.6	189.182
3000	5	0.999791	177.4	302.822

Table (1) - Randomly generated symmetric cost matrices, cost range [0-n], times are for a Sun Microsystem 4/330 deskside computer. In all cases $\lambda = 200$.

Problems from TSPLIB			
problem name	size	type	solution time (sec)
gr17.tsp	17	-	25.5
gr21.tsp	21	-	0.06
gr24.tsp	24	-	0.14
bayg29.tsp	29	Euclidean	1.26
bays29.tsp	29	Euclidean	1.99
dantzig42.tsp	42	-	440
gr48.tsp	48	-	7613
att48.tsp	48	Euclidean	-
hk48.tsp	48	-	47.79
eil51.tsp	51	Euclidean	3.16
st70.tsp	70	-	-
pr76.tsp	76	Euclidean	-
eil76.tsp	76	Euclidean	10.42
gr96.tsp	96	Euclidean	-
rat99.tsp	99	Rattled Grid	288
eil101.tsp	101	Euclidean	3332

Table (2) - These problems belong to TSPLIB, a collection of test instances for the traveling salesman and vehicle routing problems. TSPLIB contains symmetric TSP instances with up to 11849 cities and is available via anonymous ftp on the academic computer network. A time entry of - indicates the algorithm did not complete.

References

1. D. L. Miller and J. F. Pekny, "Results From A Parallel Branch and Bound Algorithm For Solving Large Asymmetric Traveling Salesman Problems," *Operations Research Letters*, vol. 8, pp. 129-135, 1989.

2. J. F. Pekny and D. L. Miller, "A Parallel Branch and Bound Algorithm For Solving Large Asymmetric Traveling Salesman Problems," *Mathematical Programming*. in press.

3. D. L. Miller and J. F. Pekny, "Exact Solution of Large Asymmetric Traveling Salesman Problems," *Science*, vol. 251, pp. 754-761, 1991.

4. M. Padberg and G. Rinaldi, "Optimization of a 532-City Symmetric Traveling Salesman Problem By Branch and Cut," *Operations Research*, vol. 6, no. 1, pp. 1-7, 1987.

5. M. Bellmore and J. C. Malone, "Pathology of Traveling Salesman Subtour-Elimination Algorithms," *Operations Research*, vol. 19, pp. 278-307, 1971.

6. M. W. Padberg and M. R. Rao, "Odd Minimum Cut-Set and b-Matchings," *Mathematics of Operations Research*, vol. 7, no. 1, pp. 67-80, 1982.

7. J. Edmonds, "Maximum Matching and a Polyhedron with 0,1-Vertices," *Journal of Research of the National Bureau of Standards*, vol. 69B , no. 1 and 2, pp. 125-130, 1965.

8. J. Edmonds and E. L. Johnson, "Matching: A Well Solved Class of Integer Linear Programs," in *Combinatorial Structures and Their Applications*, pp. 89-92, Gordon & Breach, New York, 1970.

9. R. P. Anstee, "A Polynomial Algorithm for b-Matchings: An Alternative Approach," *Information Processing Letters*, vol. 24, pp. 153-157, 1987.

10. R. P. Anstee, "An Algorithmic Proof of Tutte's f-Factor Theorem," *Journal of Algorithms*, vol. 6 , pp. 112-131, 1985.

11. U. Derigs, "Solving Non-Bipartite Matching Problems Via Shortest Path Techniques," *Annals of Operations Research*, vol. 13, pp. 225-261, 1988.

12. G. Carpaneto and P. Toth, "Some New Branching and Bounding Criteria for the Asymmetric Travelling Salesman Problem," *Management Science*, vol. 26, no. 7, pp. 736-743, 1980.

13. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, *The Traveling Salesman Problem A Guided Tour of Combinatorial Optimization*, Wiley, New York, 1985.

14. R. M. Karp, "A Patching Algorithm for the Nonsymmetric Travelling-Salesman Problem," *Siam J. Computers*, vol. 8, no. 4, pp. 561-573, 1979.

15. M. Grötschel and O. Holland, "Solution of Large-Scale Symmetric Travelling Salesman Problems," Report No. 73, Institut für Mathematik, Universität Augsburg, 1988.

16. For more information, send electronic mail to cprc@rice.edu.